International Journal of Data Science and Analytics (2024) 18:139–155
https://doi.org/10.1007/s41060-023-00428-2

REGULAR PAPER

Graph-based feature extraction on object-centric event logs

Alessandro Berti[1] · Johannes Herforth[1] · Mahnaz Sadat Qafari[1] · Wil M. P. van der Aalst[1]

Received: 28 February 2023 / Accepted: 2 July 2023 / Published online: 20 July 2023

Abstract
Process mining techniques have proven crucial in identifying performance and compliance issues. Traditional process mining, however, is primarily case-centric and does not fully capture the complexity of real-life information systems, leading to a growing interest in object-centric process mining. This paper presents a novel graph-based approach for feature extraction from object-centric event logs. In contrast to established methods for feature extraction from traditional event logs, object-centric logs present a greater challenge due to the interconnected nature of events related to multiple objects. This paper addresses this gap by proposing techniques and tools for feature extraction specifically designed for object-centric event logs. In this work, we focus on features pertaining to the lifecycle of the objects and their interaction. These features enable a more comprehensive understanding of the process and its inherent complexities. We demonstrate the applicability of our approach through its implementation in two significant areas: anomaly detection and throughput time prediction for objects in the process. Our results, based on four problems in a Procure-to-Pay process, affirm the potential of our proposed features in enhancing the scope of process mining. By effectively transforming object-centric event logs into numeric vectors, we pave the way for the application of a broader range of machine learning techniques, such as classification, prediction, clustering, and anomaly detection, thereby extending the capabilities of process mining.

Keywords Object-centric process mining · Object-based graphs · Object-centric feature extraction · Object-centric machine learning

1 Introduction

Process mining provides an established set of tools and techniques to analyze and obtain insights into the execution of an organizational business process starting from an event log. Event logs are extracted from the information systems supporting the execution of such processes.

Among existing process mining techniques, we also find applications of machine learning, including predictive analytics (e.g., predicting the remaining time of a case), prescriptive analytics (e.g., identifying the activities that should be performed to complete the case successfully), clustering (i.e., identifying groups of cases with similar behavior), and anomaly detection (i.e., identifying cases with exceptional behavior in comparison to the mainstream cases). Any application of the aforementioned techniques starts with the extraction of numeric features related to the process executions from the event log. When working with "traditional" event logs, for which a case notion is identified,[1] feature extraction associates a numeric vector to each case of the event log. The vector can contain information such as the number of occurrences per activity, the frequency/performance of the directly-follows relationships, and the values of the events' numeric attributes. The feature extraction depends on the situation. For example, if a predictor is built to predict the choice made in a decision point, then the numeric vector is extracted from the part of the trace that happened (has been recorded) before that choice point.

✉ Alessandro Berti
a.berti@pads.rwth-aachen.de

Johannes Herforth
johannes.herforth@rwth-aachen.de

Mahnaz Sadat Qafari
m.s.qafari@pads.rwth-aachen.de

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Process and Data Science Group, RWTH Aachen University, Ahornstrasse 55, 52074 Aachen, NRW, Germany

[1] A case notion is a criterion used to group events belonging to the same process execution. For example, in a ticketing management system, the identifier of the ticket can be used to group all the events that are needed to resolve the ticket.

Table 1 Example object-centric event log of a purchase requisition process represented as a table

Ev.ID	Activity	Timestamp	Purch. req.	Purch. ord.	Goods issues	Invoices	Payments
e1	Create purchase requisition	2021-03-20 10:30	['PR1']				
e2	Close purchase requisition	2021-03-20 14:00	['PR1']				
e3	Create purchase requisition	2021-03-21 09:30	['PR2']				
e4	Create purchase order	2021-03-22 14:59	['PR2']	['PO1']			
e5	Invoice receipt	2021-03-25 11:00		['PO1']		['R1']	
e6	Perform payment	2021-03-30 11:58				['R1']	['P1']
e7	Create purchase requisition	2021-04-01 09:15	['PR3']				
e8	PR formal approval	2021-04-01 10:15	['PR3']				
e9	Create purchase order	2021-04-02 17:00	['PR3']	['PO2']			
e10	Change purchase requisition	2021-04-03 10:00	['PR3']				
e11	Invoice receipt	2021-04-05 15:00		['PO2']		['R2']	
e12	Perform payment	2021-04-15 09:27				['R2']	['P2']
e13	Create purchase order	2021-04-17 14:29		['PO3']			
e14	Invoice receipt	2021-04-28 10:00		['PO3']		['R3']	
e15	Perform payment	2021-04-30 15:00				['R3']	['P3']
e16	Invoice receipt	2021-05-28 10:01		['PO3']		['R4']	
e17	Perform payment	2021-05-30 15:17				['R4']	['P4']
e18	Invoice receipt	2021-06-28 10:01		['PO3']		['R5']	
e19	Perform payment	2021-06-30 15:29				['R5']	['P5']
e20	Create purchase requisition	2021-07-01 11:15	['PR4']				
e21	Create purchase order	2021-07-02 09:38	['PR4']	['PO4']			
e22	Invoice receipt	2021-07-09 16:00		['PO4']		['R6']	
e23	Goods issue	2021-07-11 10:30		['PO4']	['GI1']		
e24	Perform payment	2022-05-15 09:00				['R6']	['P6']
e25	Invoice receipt	2022-05-20 12:00				['R7']	
e26	Create purchase order	2022-05-20 15:00		['PO5']		['R7']	
e27	Create purchase order	2022-06-01 09:17		['PO6']			
e28	Create purchase order	2022-06-02 11:48		['PO7']			
e29	Create invoice	2022-06-05 09:00		['PO6', 'PO7']		['R8']	

In real-life information systems, the assumption that an event is related to a single case/object is unrealistic. For example, events in ERP systems can be related to several objects of different object types (e.g., an event of invoice creation is related to an object of type "invoice" and potentially many objects of type "order"). Object-centric process mining develops on these assumptions. Object-centric process mining techniques require object-centric event logs (OCELs, [11]). An example of an OCEL in tabular form is contained in Table 1. For example, the first row contains the event with identifier *e1*, activity *Create Purchase Requisition*, and timestamp *2021-03-20 10:30*. This event is related to a single object *PR1* of type *Purch.Req.*

Object-centric event logs, given the possibility to relate multiple objects to an event, help to resolve the *convergence* and *divergence* issues [24] observed in traditional event logs. A convergence issue arises when an event (for example, the creation of an invoice) needs to be replicated in different cases (for example, all the orders cited in the invoice). A divergence issue arises when several instances of the same activity (for example, the receipt of an invoice) are contained in the same case, describing a non-existing loop. In the development of object-centric process mining techniques, both the lifecycle (sequence of related events) of the objects and their interactions can be considered. Considering both aspects, it is possible to extract more meaningful numeric features for the application of machine learning techniques.

In this paper, we propose an approach for feature extraction on object-centric event logs that can consider traditional (related to the events of the lifecycle of an object) and graph-based (related to different types of interactions between objects) features. Two tools (pm4py and OCPM) are offered to extract the features from an object-centric event log and

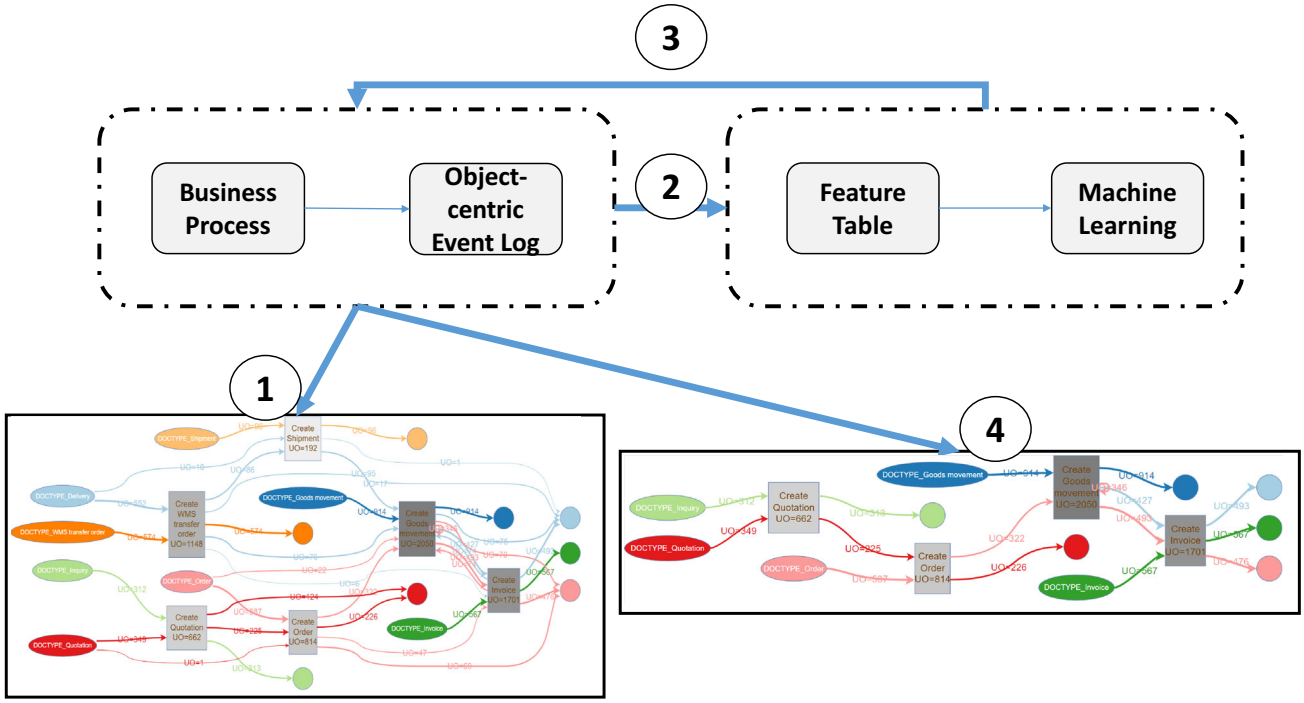

Fig. 1 Summary of the approach described in the paper. Starting from a business process, we could extract an object-centric event log from the information system supporting the process. A model describing the executions (1) of the process could be quite complex. In this paper, we describe an approach to enable feature extraction on object-centric event logs, which allows for the application of machine learning techniques (2) to improve the process (3) and eventually streamline its execution (4), making the process simpler and more effective

use the features to improve the execution of the business processes. The following application areas are proposed:

- Checking conformance patterns in the object-centric event logs (e.g., an invoice should always be preceded by an order).
- Identifying and filtering the outlier behavior to focus on the mainstream/exceptional behavior (e.g., mainstream behavior to discover more meaningful process models, and exceptional behavior for diagnostics purposes).
- Visualize the correlation between different variables of the process to understand the process (e.g., is a high number of open cases affecting negatively the throughput time of the process?).

A real-life application to a Procure-to-Pay (P2P) process is proposed, in which three conformance aspects (identification of maintenance contracts, maverick buying, and post-mortem changes to purchase requisitions) are verified on top of the object-centric event log of the process.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces the essential definition of object-centric event log. Section 4 describes the extraction of object-related graphs from the event log, their conversion to numeric feature maps, and the propagation of the values of these features to neighboring objects. Section 5 presents two tools performing object-centric feature extraction. Section 6 assesses graph-based feature extraction in its ability to identify several real-life problems in a Procure-to-Pay process. Finally, Sect. 7 concludes the paper.

2 Related work

In this section, we present the relevant work on feature extraction, and its applications, on traditional or object-centric event logs.

Graph-Based Analysis of Event Data: The graph-based nature of event data has been described in [9]. Here, a graph database (Neo4J) encodes the events and attributes as nodes of a graph. Derived structures (for example, the directly-follows graph given the lifecycle of an object) are computed directly on top of the graph database (using the Cypher language). The advantage of using graph databases is their scalability in the querying/analysis of complex relationships between the different nodes of the graph. Applications to publicly available logs are described throughout the paper.

Feature Extraction on Traditional Event Logs: A feature extraction approach starting from traditional event logs is described in [6]. In this framework, each case is associated

with a numeric vector describing the behavior observed in the case (for example, the number of occurrences for an activity, the paths' frequency/performance, the duration, and the values of numeric/string attributes). Depending on the target application, the numeric vector can be different (situation). For example, if the goal is to build a predictor of the total time of the case starting from the knowledge of the first event, only the features related to the attributes of the first event can be considered. In [8], additional features helpful for machine learning purposes are engineered starting from process models automatically discovered from the event logs.

In [14, 18], inter-case features are discovered from traditional event logs to improve predictive tasks (such as the prediction of the remaining time). of the remaining time). An example of an inter-case feature is batching, in which lots of cases are waiting for closure. However, this only occurs at given points of time or when enough cases are in the queue. Also, instance-spanning constraints [26] are discovered from event logs discovered advanced inter-case correlations (an example of ISC follows: a washing machine can start only when there are at least 5 dresses inside it). Inter-case features can be seen as object-to-object relationships. In our approach, we considered for example the work-in-progress metric. In [5], features that are useful for prediction are extracted from the performance spectrum.

Temporal features (inspired by the system dynamics paradigm) can be extracted from event logs which are useful to build simulation models [16] and therefore supporting decisions in production line processes [17]. These are computed by dividing the time interval into subintervals, computing metrics such as the number of events/resources/activities, the case arrival/finish rate, the service/waiting time, and computing linear relations over them.

Graph embeddings have been used to encode event graphs and compute high-quality features [13]. In [25], different encoding techniques have been used to enrich an ongoing process execution with information related to advanced control-flow patterns.

There are several works that propose feature extraction methods with specific goals. For example, in [19], a feature extraction method has been explained and then the problem of discovering potential causal relationships between a set of extracted features has been addressed. This helps to find problematic features of the process and estimate the effect of an intervention on such features. In [20], results from causal inference are adapted on top of features calculated on top of event logs to be able to reason over event logs and process interventions. Furthermore, causal inference methods are adapted on top of the features calculated from an event log, allowing reasoning about the event log and process interventions.

Feature Extraction on Object-Centric Event Logs: In [1], the interconnections between different objects in an object-centric event log are used to define variants in the object-centric setting. In particular, the connected components of the object interaction graph are used to identify clusters of related objects. These clusters are used to identify correlated events in the object-centric event log. The graph-related properties of these events and objects are then used to identify events/objects with similar behavior (this is eventually called variant). These properties are also exploited in [2] for feature extraction on object-centric event logs. Different use cases (including predictive analytics) and types of feature extraction (table-based, graph-based) are proposed. In particular, the importance of the single features for the target is identified using explainable AI techniques (including the usage of SHAP values). The assessment of the paper is done starting from a traditional event log "adapted" as an object-centric event log. The related tool support is publicly available and can be easily installed. The paper [10] introduces feature extraction on object-centric event logs for the goals of predictive analytics (prediction of the remaining time, activities occurrences, customer satisfaction). It shows how considering the interaction between objects (object interaction graph) leads to an improvement in the quality of the prediction on object-centric event logs, compared to only considering the flattened event logs. This has been evaluated in a real-life case study. Also, [12] considers predictive monitoring starting from object-centric event logs, focusing on the prediction of the next activity and remaining time starting from traditional and interaction-based features.

A qualitative comparison with the related work highlights distinctive aspects of our approach. While [1] leverage interconnections between different objects for feature extraction, it does not distinguish different types of interactions between objects, as we do. Furthermore, our work uniquely considers diverse use cases such as anomaly detection and object-centric event log querying in addition to predictive analytics. The papers [10, 12] also utilize object interactions for predictive analytics. However, they primarily focus on prediction tasks such as remaining time prediction and next activity prediction. In contrast, our approach not only facilitates predictive analytics but also significantly extends its scope by accommodating use cases like anomaly detection and event log querying. Moreover, our work provides a graphical tool to facilitate these use cases, as opposed to a library, contributing to the usability of the proposed techniques. Therefore, the versatility of our approach and its broadened focus on various types of object interactions and use cases distinguishes it qualitatively from the related works. In terms of a direct comparison on prediction accuracy or error rates (as inferred from metrics like MAPE and RMSPE), a comprehensive comparison may not be feasible due to the diverse datasets and use cases across different works. However, the broadened use cases and enhanced usability of our tool support the potential

of our approach to generate valuable insights across various process mining scenarios.

Applications of Machine Learning in Process Mining: Machine learning (ML) has become a cornerstone in the field of process mining, significantly contributing to the advancement of various aspects of this discipline. As evidenced in [22], long short-term memory (LSTM) neural networks are explored for predictive process monitoring tasks. With the capability to remember past information over long periods, LSTM presents a consistent and accurate model for prediction tasks such as identifying the next event and its timestamp in a running process case, projecting the full continuation of a case, and estimating the remaining time of the process. As these tasks are central to process monitoring, the application of LSTM suggests a promising enhancement in performance over traditional methods.

The paper [23] delves into the intersection of ML and process mining, examining the role of sequence modeling methods in predicting the next element in a sequence. Machine learning methods, such as (hidden) Markov models and recurrent neural networks, often demonstrate superior accuracy in capturing sequential behavior in data compared to traditional process mining and grammar inference fields. This superiority suggests their potential for broader applications within the realm of process mining.

In [7], machine learning is harnessed for anomaly detection in logs of process-aware information systems (PAISs). Given the inherent risks in businesses such as financial losses due to mismanagement, anomaly detection serves a critical role in ensuring governance best practices and operational security. The integration of ML-powered tools like ProM for anomaly detection in process logs can enhance the capabilities of PAISs, creating a more robust, secure, and competitive business environment.

The paper [21] underscores a significant challenge in process mining: the concept drift. Machine learning techniques are instrumental in detecting and adapting to concept drift in evolving environments, facilitating online process mining in contrast to the traditional offline analysis. However, the study also highlights the need for common evaluation protocols, datasets, and metrics for a more effective assessment of these ML techniques, which would further bolster their applications in handling concept drift within process mining.

3 Preliminaries

In the following, we will introduce the definition of object-centric event log, which is the core concept on top of which the filters will be applied. In Definition 1, some universes needed for the definition of OCEL are introduced.

Definition 1 *(Universes (for OCEL))* Some universes are used in the formal definition of object-centric event logs are: U_Σ is the universe of all the strings; U_e is the universe of event identifiers; U_{act} is the universe of activities; U_{timest} is the universe of timestamps; U_{att} is the universe of attribute names; U_{val} is the universe of attribute values; U_{typ} is the universe of attribute types; U_o is the universe of object identifiers; U_{ot} is the universe of objects types.

Definition 2 introduces the formal definition of object-centric event log.

Definition 2 *(Object-Centric Event Log)* An object-centric event log is a tuple $L = (E, AN, AV, OT, O, \pi_{\text{act}}, \pi_{\text{time}}, \pi_{\text{vmap}}, \pi_{\text{omap}}, \pi_{\text{otyp}}, \pi_{\text{ovmap}}, \leq)$ such that:

- $E \subseteq U_e$ is the set of event identifiers.
- $AN \subseteq U_{\text{att}}$ is the set of attributes names.
- $AV \subseteq U_{\text{val}}$ is the set of attribute values (with the requirement that $\text{AN} \cap \text{AV} = \emptyset$).
- $OT \subseteq U_{\text{ot}}$ is the set of object types.
- $O \subseteq U_o$ is the set of object identifiers.
- $\pi_{\text{act}} : E \rightarrow U_{\text{act}}$ is the function associating an event (identifier) to its activity.
- $\pi_{\text{time}} : E \rightarrow U_{\text{timest}}$ is the function associating an event (identifier) to a timestamp.
- $\pi_{\text{vmap}} : E \rightarrow (AN \nrightarrow AV)$ is the function associating an event (identifier) to its attribute value assignments.
- $\pi_{\text{omap}} : E \rightarrow \mathcal{P}(O)$ is the function associating an event (identifier) to a set of related object identifiers.
- $\pi_{\text{otyp}} : O \rightarrow OT$ assigns precisely one object type to each object identifier.
- $\pi_{\text{ovmap}} : O \rightarrow (AN \nrightarrow AV)$ is the function associating an object to its attribute value assignments.
- $\leq$ is a total order on the events (i.e., it respects the antisymmetry, transitivity, and connexity properties). A possible way to define a total order is to consider the timestamps associated with the events as a pre-order (i.e., assuming some arbitrary, but fixed, order for events having the same timestamp).

To explain better the definition, we can map the entities in the definition to the example in Table 1. In this case:

- $E = \{e1, e2, \ldots, e29\}$.
- $OT = \{$"Purch.Req", "Purch.Ord", "Goods Issues", "Invoices", "Payments"$\}$.
- $O = \{PR1, PR2, PR3, PR4, PO1, PO2, PO3, PO4, PO5, PO6, PO7, GI1, R1, R2, R3, R4, R5, R6, R7, R8, P1, P2, P3, P4, P5, P6\}$.
- $\pi_{\text{act}}(e1) =$ "Create Purchase Requisition".
- $\pi_{\text{act}}(e_2) =$ "Close Purchase Requisition".
- $\pi_{\text{time}}(e1) =$ 2021-03-20 10:30, $\pi_{\text{time}}(e2) =$ 2021-03-20 14:00, ...

- $\pi_{\text{omap}}(e1) = \{PR1\}, \pi_{\text{omap}}(e4) = \{PR2, PO1\}$.
- $\pi_{\text{otyp}}(PR1) =$ "Purch.Req", $\pi_{\text{otyp}}(PO1) =$ "Purch.Ord", $\pi_{\text{otyp}}(R1) =$ "Invoices", ...
- $e1 \leq e2 \leq e3 \leq \cdots \leq e29$

In Definition 2, every event is related through π_{omap} to several objects of different object types. The reverse is also true: An object can be related to a sequence of events in the object-centric event log. This is called *lifecycle* and is introduced in Definition 3.

Definition 3 *(Lifecycle of an Object)* Let L be an object-centric event log as in Definition 2. Given an object $o \in O$, we define $\text{lif}(o) = \langle e_1, \ldots, e_n \rangle$ as the ordered sequence of events $e_1, \ldots, e_n \in E$ which are related to the object, i.e., $\forall i \in \{1, \ldots, n\} : o \in \pi_{\text{omap}}(e_i)$. We define the start event $\text{start}(o) = \text{lif}(o)(1)$ and the end event $\text{end}(o) = \text{lif}(o)(|\text{lif}(o)|)$.

Looking to the object-centric event log described in Table 1, we could see that the lifecycle of PR1 is the sequence of events $\langle e1, e2 \rangle$, while the lifecycle of $PO4$ is the sequence of events $\langle e21, e22, e23 \rangle$. Consequently, $\text{start}(PR1) = e1$, $\text{end}(PR1) = e2$, and $\text{start}(PO4) = e21$, $\text{end}(PO4) = e23$.

Recently, the OCEL standard http://www.ocel-standard.org/ has been introduced for the storage of object-centric event logs [11]. Two different implementations of the standard are provided (JSON-OCEL and XML-OCEL), which are based on popular file formats.

Object-centric process mining is also supported by commercial software. For example, the Celonis vendor recently introduced the *Process Sphere* feature, which allows "to analyze and visualize the complex relationships between events and objects across interconnected processes"[2] and can ingest object-centric event logs in the OCEL standard.

4 Approach

In this section, we want to define graph-related features on top of object-centric event logs. Since real-life processes include many-to-many interactions between objects, this can be useful for detecting several problems (for example, by looking if an expected relation between some objects is recorded in the object-centric event log). In the assessment of the paper (Sect. 6), we will show the application of these features extracted from an object-centric event log of a P2P (Procure-to-Pay) process for detecting problems such as maverick buying (the order is placed without approval, so the invoice is received from the company without a pre-existing purchase order in the system), post-mortem changes to the purchase requisitions and the detection of maintenance contracts as a special category of purchase orders.

The section is organized as follows. In Sect. 4.1, we introduce the definition of object-based graph and three important object-based graphs. In Sect. 4.2, we introduce the definition of object-based feature maps and introduce two examples (basic and graph-based). In Sect. 4.3, the concept of feature propagation is introduced which helps in understanding the influence between the objects. In Sect. 4.4, an approach for the automatic identification of anomalous conditions is proposed.

4.1 Object-based graphs

An object-centric graph describes the relationships between the objects of an object-centric event log. This helps to identify several patterns of interest. In Definition 4, the generic definition of an object-based graph is proposed.

Definition 4 *(Object-Based Graph)* Given a set of objects $O \subseteq U_O$ and $\Lambda \subseteq O \times O$, (O, Λ) is an object-based graph.

Some important object-based graphs are presented in Definition 5.

Definition 5 *(Object Interaction, Creation, and Continuation Graph)* Let L be an object-centric event log as in Definition 2. We define the following object-based graphs:

1. *Object Interaction Graph*: $G = (O, \Lambda), \Lambda = \cup_{e \in E}\{(o_1, o_2) \mid o_1, o_2 \in \pi_{\text{omap}}(e) \wedge o_1 \neq o_2\}$.
2. *Object Creation Graph*: $G = (O, \Lambda), \Lambda = \cup_{e \in E}\{(o_1, o_2) \mid o_1, o_2 \in \pi_{\text{omap}}(e) \wedge e \neq \text{start}(o_1) \wedge e = \text{start}(o_2)\}$
3. *Object Continuation Graph*: $G = (O, \Lambda), \Lambda = \cup_{e \in E}\{(o_1, o_2) \mid o_1, o_2 \in \pi_{\text{omap}}(e) \wedge e \neq \text{start}(o_1) \wedge e = \text{end}(o_1) \wedge e = \text{start}(o_2)\}$

In Definition 5, the *object interaction graph* is introduced, connecting every couple of objects co-occurring in the set of related objects of an event in the event log. The nodes of this graph are the objects, and the set of edges Λ contains pairs of objects. The relation is symmetric, which means that for all the pairs of objects $o_1, o_2 \in O$, $(o_1, o_2) \in \Lambda \iff (o_2, o_1) \in \Lambda$. Figure 2 shows an example of an object interaction graph computed on top of the event log described in Table 1. Given the symmetry of the relation, the graph has been represented as an undirected graph. In particular, since in Table 1 the objects $PO6$, $PO7$, and $R8$ are all related to the event $e29$, they are connected.

The object interaction graph can be used to see whether an expected relation exists. For example, we could expect that every purchase order in a P2P system is connected to at

[2] https://www.celonis.com/blog/celonis-announces-next-generation-mri-process-mining-technology-with-process-sphere/.

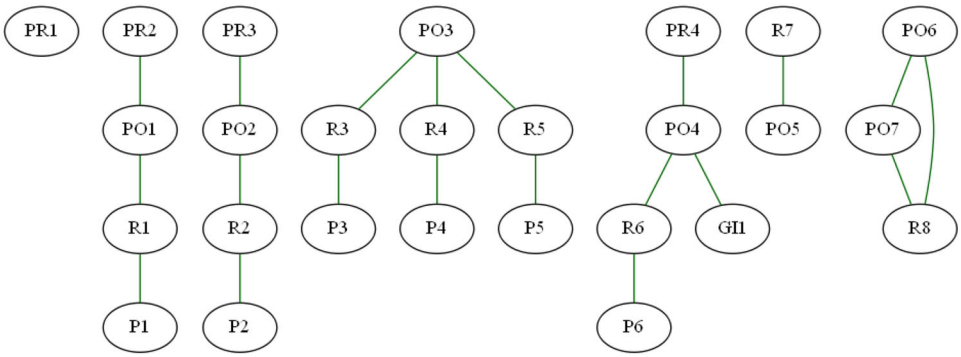

Fig. 2 Object interaction graph built on top of the event log described in Table 1

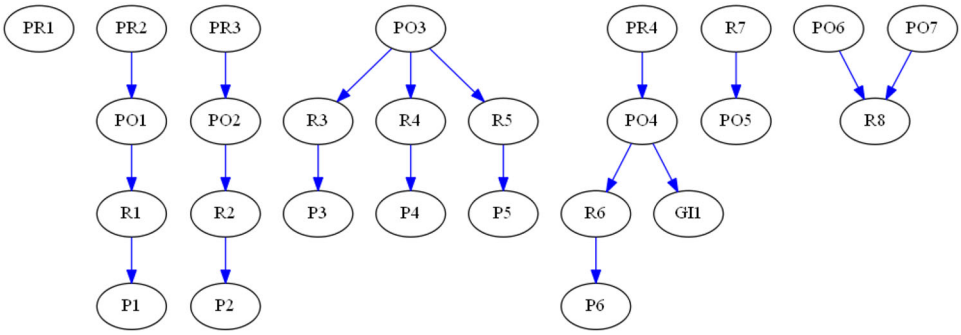

Fig. 3 Object creation graph built on top of the event log described in Table 1

least one purchase requisition (which implies that the order followed a standard approval process).

The *object creation graph* is also introduced in Definition 5. This graph considers the sequence of the events of the log and, for every event, connects every pre-existing object (i.e., an object started before the current event) to every novel object (whose lifecycle starts with the current event). Looking at Table 1, $PR2$ (starting its lifecycle in $e3$ and belonging to the related objects of the event $e4$) is connected in the object creation graph to $PO1$ (starting its lifecycle in $e4$). Figure 3 shows an example of an object creation graph computed on top of the event log described in Table 1. In this case, the relation is asymmetric, and we can represent a directed graph. We note that, differently from the object interaction graph shown in Fig. 2, $PO6$ and $PO7$ are not directly connected, but they are connected by an arc to $R8$ (since $R8$ starts its lifecycle in $e29$ and both $PO6$ and $PO7$ start their lifecycle earlier and end it at $e29$).

The object creation graph can be used to see whether the relations between the objects follow the expected temporal order. For example, in a P2P system, we can expect an order to be followed by the invoice and not vice versa (this problem is called maverick buying).

In Definition 5, also the *object continuation graph* is introduced. This graph connects two objects $o_1, o_2 \in O$ when the lifecycle of $o1$ is terminated by the same event in which the lifecycle of $o2$ starts. Looking at Table 1, we see that the event $e4$ terminates the lifecycle of the purchase requisition $PR2$ and starts the lifecycle of the purchase order $PO1$. Hence, $PR2$ and $PO1$ are connected in the object continuation graph. Figure 4 shows an example of an object continuation graph computed on top of the event log described in Table 1. In this case, the relation is asymmetric, and we can represent a directed graph. We note that, differently from the object creation graph shown in Fig. 3, the purchase requisition $PR3$ is not connected to the purchase order $PO2$, because the purchase requisition is changed after the start of the lifecycle of $PO2$. Note that the object continuation graph contains a subset of the edges contained in the object creation graph.

The object continuation graph can be used to verify whether the lifecycle of some objects (e.g., the purchase requisitions) is terminated when the lifecycle of some other objects starts (for example, the purchase orders).

4.2 Object-based feature maps

An object-based feature map associates each object of an object-centric event log to a set of numeric features. This is essential to perform any machine learning task (classification, prediction, anomaly detection, etc.).

Definition 6 (Object-Based Feature Map) Given a set of objects $O \subseteq U_o$ and a set of features $\Sigma \subseteq U_\Sigma$, an object-based feature map is a function $O \to (\Sigma \to \mathbb{R})$.

Definition 7 associates each object to some basic features: the number of related events, the throughput time as the difference between the timestamps of the last and first event of the lifecycle, the work-in-progress metric, which counts

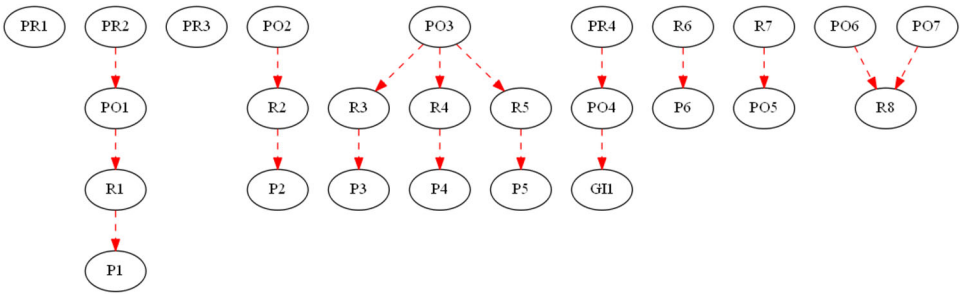

Fig. 4 Object continuation graph built on top of the event log described in Table 1

how many objects have a lifecycle with (non-empty) temporal overlap with the lifecycle of the current event, and for every activity of the log, the number of occurrences in the lifecycle of the object. Note that many other features could be easily introduced (for example, based on the paths' frequency or performance) but are not included for simplicity in Definition 7.

Definition 7 *(Basic Feature Map)* Let L be an object-centric event log as in Definition 2. Given $\text{ACT} = \{\pi_{\text{act}}(e) \mid e \in E\}$, and $\Sigma = \{\text{“}relevs\text{”}, \text{“}throughput\text{”}, \text{“}wip\text{”}\} \cup \{\text{“\#”}@a \mid a \in \text{ACT}\}$ (where @ is the concatenation of strings), we define the basic feature map $f_{\text{bas}} : O \to (\Sigma \to \mathbb{R})$ in which:

- for $o \in O$, $f_{\text{bas}}(o)(\text{“relevs”}) = |\text{lif}(o)|$.
- for $o \in O$, $f_{\text{bas}}(o)(\text{“throughput”}) = \pi_{\text{time}}(\text{end}(o)) - \pi_{\text{time}}(\text{start}(o))$.
- for $o \in O$, $f_{\text{bas}}(o)(\text{“wip”}) = |\{o' \in O \mid [\pi_{\text{time}}(\text{start}(o)), \pi_{\text{time}}(\text{end}(o))] \cap [\pi_{\text{time}}(\text{start}(o')), \pi_{\text{time}}(\text{end}(o'))] \neq \emptyset\}|$.
- for $o \in O$ and $a \in \text{ACT}$, we define $f_{\text{bas}}(o)(\text{“\#”}@a) = |\{e \in \text{lif}(o) | \pi_{\text{act}}(e) = a\}|$.

Looking at the object-centric event log proposed in Table 1, and focusing on the object $PO4$, we have:

- $f_{\text{bas}}(PO4)(\text{“}relevs\text{”}) = 3$ (because the events $e21$, $e22$ and $e23$ are related to $PO4$).
- $f_{\text{bas}}(PO4)(\text{“}throughput\text{”}) \approx 9\text{days}$ (its lifecycle starts in $e21$ and terminates in $e23$).
- $f_{\text{bas}}(PO4)(\text{“}wip\text{”}) = 4$ (because the objects $PR4$, $GI1$ and $R6$ have temporal intersection with its lifecycle, and also $PO4$ itself).
- $f_{\text{bas}}(PO4)(\text{“\#}CreatePurchaseOrder\text{”}) = 1$ (because there is a single occurrence of the activity *Create Purchase Order* for the object).

The basic feature map on top of the object-centric event log proposed in Table 1 is presented in Table 2.

The calculation of graph-based features is introduced in Definition 8. More comprehensive feature maps using the contributions of different graphs could be defined. For example, the structural properties of the vertices of the graph can be exploited. However, this is not done for simplicity in Definition 8.

Definition 8 *(Graph-Based Feature Map)* Let L be an object-centric event log as in Definition 2 and $G = (O, \Lambda)$ an object-based graph. Given $\Sigma = \{\text{“}outdegree\text{”}\} \cup \{\text{“}outdegreeot\text{”}@ot \mid ot \in OT\}$ (where @ is the concatenation of strings), we define the feature map $f_{\text{graph}} : O \to (\Sigma \to \mathbb{R})$ in which:

- for $o \in O$, $f_{\text{graph}}(o)(\text{“outdegree”}) = |\{(o_1, o_2) \in \Lambda \mid o_1 = o\}|$
- for $o \in O$ and $ot \in OT$, $f_{\text{graph}}(o)(\text{“outdegreeot”}@ot) = |\{(o_1, o_2) \in \Lambda \mid o_1 = o \wedge \pi_{\text{otyp}}(o_2) = ot\}|$.

The feature map proposed in Definition 8 includes both the number of outgoing connections of the object to the other objects and the number of connected objects per object type. Looking at the object-centric event log in Table 1, if we choose the object interaction graph (represented in Fig. 2) and focus on the object $PO4$, we have:

- $f_{\text{graph}}(PO4)(\text{“}outdegree\text{”}) = 3$ (because the objects $PR4$, $GI1$ and $R6$ are connected to $PO4$ in the object interaction graph).
- $f_{\text{graph}}(PO4)(\text{“}outdegreeot Purch.Req.\text{”}) = 1$ (because the purchase requisition $PR4$ is connected to $PO4$).
- $f_{\text{graph}}(PO4)(\text{“}outdegreeot Invoices\text{”}) = 1$ (because the invoice $R6$ is related to $PO4$).
- $f_{\text{graph}}(PO4)(\text{“}outdegreeot GoodsIssues\text{”}) = 1$ (because the goods issue $GI1$ is connected to $PO4$).

The graph-based feature map computed on the object-centric event log in Table 1 and its object interaction graph is proposed in Table 3.

4.3 Feature propagation

We introduce in Definition 9 the *feature propagation* approach to "propagate" the values of the features of the neighboring objects.

Table 2 Basic feature map on top of the object-centric event log proposed in Table 1

ID	Relevs	Throughput	wip	#Change PR	#Close PR	#Cr. Inv.	#Cr. PO	#Cr. PR	#G. Issue	#Inv. Receipt	#PR Appr.	#Payment
PR1	2	12600.0	1	0	1	0	0	1	0	0	0	0
PR2	2	106140.0	2	0	0	0	1	1	0	0	0	0
PR3	4	175500.0	2	1	0	0	1	1	0	0	1	0
PR4	2	80580.0	2	0	0	0	1	1	0	0	0	0
PO1	2	244860.0	3	0	0	0	1	0	0	1	0	0
PO2	2	252000.0	3	0	0	0	1	0	0	1	0	0
PO3	4	6204720.0	6	0	0	0	1	0	0	3	0	0
PO4	3	780720.0	4	0	0	0	1	0	1	1	0	0
PO5	1	0.0	2	0	0	0	1	0	0	0	0	0
PO6	2	344580.0	3	0	0	1	1	0	0	0	0	0
PO7	2	249120.0	3	0	0	1	1	0	0	0	0	0
R1	2	435480.0	3	0	0	0	0	0	0	1	0	1
R2	2	844020.0	3	0	0	0	0	0	0	1	0	1
R3	2	190800.0	3	0	0	0	0	0	0	1	0	1
R4	2	191760.0	3	0	0	0	0	0	0	1	0	1
R5	2	192480.0	3	0	0	0	0	0	0	1	0	1
R6	2	26758800.0	4	0	0	0	0	0	0	1	0	1
R7	2	10800.0	2	0	0	0	1	0	0	1	0	0
R8	1	0.0	3	0	0	1	0	0	0	0	0	0
GI1	1	0.0	3	0	0	0	0	0	1	0	0	0
P1	1	0.0	2	0	0	0	0	0	0	0	0	1
P2	1	0.0	2	0	0	0	0	0	0	0	0	1
P3	1	0.0	3	0	0	0	0	0	0	0	0	1
P4	1	0.0	3	0	0	0	0	0	0	0	0	1
P5	1	0.0	2	0	0	0	0	0	0	0	0	1
P6	1	0.0	2	0	0	0	0	0	0	0	0	1

Definition 9 *(Feature Propagation)* Let L be an object-centric event log as in Definition 2, $G = (O, \Lambda)$ an object-based graph and $f : O \to (\Sigma \to \mathbb{R})$ an object-based feature map. Given an aggregation function $\text{agg} : \mathcal{P}(\mathbb{R}) \to \mathbb{R}$, we define $f_{\text{agg},G} : O \to (\Sigma \to \mathbb{R})$ such that for $\sigma \in \Sigma$ and $o \in O$, $f_{\text{agg},G}(o)(\sigma) = \text{agg}(\{f(o')(\sigma) \mid o = o' \vee (o, o') \in \Lambda\})$.

An example application is the following. Looking at the object-centric event log proposed in Table 1, if we choose the object interaction graph (represented in Fig. 2) and focus on the invoice $R6$, we can see that the invoice is connected to the purchase order $PO4$. We can see also that $PO4$ is connected to the goods issue $GI1$. The presence of a goods issue connected to $PO4$ could significantly delay the payment of the invoice $R6$. Hence, the number of goods issues connected to a purchase order is a relevant feature also for the invoice, and the feature propagation introduced in Definition 9 finds a good application in this context.

4.4 Automatic identification of anomalous conditions

Once a feature map (as in Definition 7) is defined, we can identify the objects with anomalous values in the features, or strange combinations of these values, using any anomaly detection method (for example, isolation forests [15]). Given an object-centric event log (as in Definition 2) and any $ot \in OT$, we consider the set of objects $O_{\text{ot}} = \{o \in O \mid \pi_{\text{otyp}}(o) = \text{ot}\}$ having object type ot. An anomaly detection method would split this set of objects into a set NO_{ot} (objects with normal behavior) and AO_{ot} (objects with anomalous behavior). We are interested in explaining the decisions of the anomaly detection method and consequently understanding the differences in the values of the features between the objects classified as NO_{ot} and the objects classified as AO_{ot}. This can be done using different approaches, for example using decision trees or the RIPPER [4] algorithm to get explainable classification rules describing the

Table 3 Graph-based feature map based on the object-centric in Table 1 and its object interaction graph

ID	Outdegree	Outdegree Goods issues	Outdegree Invoices	Outdegree Payments	Outdegree Purch. orders	Outdegree Purch. req.
PR1	0	0	0	0	0	0
PR2	1	0	0	0	1	0
PR3	1	0	0	0	1	0
PR4	1	0	0	0	1	0
PO1	2	0	1	0	0	1
PO2	2	0	1	0	0	1
PO3	3	0	3	0	0	0
PO4	3	1	1	0	0	1
PO5	1	0	1	0	0	0
PO6	2	0	1	0	1	0
PO7	2	0	1	0	1	0
R1	2	0	0	1	1	0
R2	2	0	0	1	1	0
R3	2	0	0	1	1	0
R4	2	0	0	1	1	0
R5	2	0	0	1	1	0
R6	2	0	0	1	1	0
R7	1	0	0	0	1	0
R8	2	0	0	0	2	0
GI1	1	0	0	0	1	0
P1	1	0	1	0	0	0
P2	1	0	1	0	0	0
P3	1	0	1	0	0	0
P4	1	0	1	0	0	0
P5	1	0	1	0	0	0
P6	1	0	1	0	0	0

differences between NO_{ot} and AO_{ot}. These rules take into account combinations of values of the features.

If we consider a Procure-to-Pay process, the following conditions can be identified automatically:

- Every purchase order should be connected to at least a purchase requisition, so if we choose the object interaction graph in Definition 8, $f_{\text{graph}}(o)(\text{“}outdegreeot Purch.Req.\text{''}) >= 1$ for the non-anomalous purchase orders.
- The orders should precede the corresponding invoices, so if we choose the object creation graph in Definition 8, $f_{\text{graph}}(o)(\text{“}outdegreeot Purch.Ord.\text{''}) = 0$ for the non-anomalous invoices.
- Purchase requisitions should be followed by at least a purchase order and should not be modified when the purchase order is created. So, if we choose the object continuation graph in Definition 8, $f_{\text{graph}}(o)(\text{“}outdegreeot Purch.Ord.\text{''}) >= 1$ for the non-anomalous purchase requisitions.
- Length and duration of the lifecycle: We can identify bounds of duration and length for the lifecycle of the objects for any object type, which can be used to identify “surprising” objects to investigate. This can be done using the $f_{\text{bas}}(o)(\text{“}throughput\text{''})$ and $f_{\text{bas}}(PO4)(\text{“}relevs\text{''})$ features introduced in Definition 7.

Some anomaly detection methods (such as isolation forests) return an anomaly score for each considered object (the higher the score, the more anomalous the object). By sorting the objects based on their anomaly score, we can consider the first objects of this list for further investigation.

5 Tool

In this section, we present two tools for object-centric process mining and machine learning on object-centric event logs. In particular, the *OCPM* web-based tool https://www.ocpm.info and the Python process mining library *pm4py* https://pm4py.fit.fraunhofer.de are described.

OCPM is an object-centric process mining tool that offers support for:

- Ingestion/exporting of object-centric event logs in the OCEL standard format (JSON-OCEL and XML-OCEL).
- Flattening the object-centric event logs into traditional event logs with the choice of a case notion.
- Advanced preprocessing features (filtering, sampling).
- Discovering object-centric process models: object-centric directly-follows graphs and object-centric Petri nets.
- Conformance checking on object-centric event logs based on declarative and temporal constraints (log skeleton, temporal profile).
- Exploration of the events/objects of the object-centric event log.

A machine learning component is available that implements the traditional and graph-based features described in this paper. With these features, different process mining applications are possible:

- With the SQL explorer, it is possible to manually explore the patterns emerging in the object-centric event log. For example, it is possible to explore when maverick buying happens in a Procure-to-Pay process (i.e., the invoice creates the order and not vice versa).
- With anomaly detection, it is possible to explore (and then filter) the objects with the most anomalous patterns. This helps to simplify the process models to the mainstream behavior.
- With correlation analytics, it is possible to understand which factors are influencing the outcome of a process. For example, the work in progress influences the duration of the invoice processing.
- With conformance checking, it is possible to explore the outliers objects for a selected feature. In this case, outlier objects can be filtered out, and this can help in reducing the process model.

pm4py is a process mining library in Python, which implements object-centric process mining feature extraction. An object-centric machine learning module is available that offers the traditional and graph-based features described in this paper. This allows for integration with the rich Python ecosystem for machine/deep learning.

6 Assessment

We divide the assessment into three sections (qualitative real-life, qualitative synthetic, and quantitative synthetic). As the real-life event log that has been used to assess the techniques is not publicly available, synthetic event logs have been provided and can be used with the tool support described in Sect. 5.

Table 4 Characteristics of the real-life event log used for the qualitative assessment

Characteristic	Value
Number of events	18037
Number of objects	9197
Number of activities	225
Number of object types	10

6.1 Qualitative assessment using a real-life P2P event log

The proposed feature extraction has been used on a real-life object-centric event log (the characteristics of this event log are described in Table 4) related to a Procure-to-Pay (P2P) process in SAP. The goal is to prove the utility of the aforementioned features to find useful insights related to the process. For each one of the proposed graphs, we found an application in the Procure-to-pay process. Moreover, feature propagation proved useful in correlating features of the purchase order to the total time of the process.

6.1.1 Application of the object continuation graph: post-mortem changes to purchase requisitions

Problem Statement: big purchase orders require, in most cases, formal approval through a purchase requisition. The payment amount/quantity of the purchase order is matched against the purchase requisition to check for possible deviations. An inconvenient step is related to the creation of a smaller purchase requisition that is formally approved and the subsequent placement of a purchase order that exceeds the amount/quantity set in the purchase requisition. The purchase requisition is eventually updated to match the purchase order. This could be easily visualized after building the object continuation graph from the object-centric event log. In this case, purchase requisitions that are disconnected from the purchase order are either:

- Not approved. An example of this is PR1 in Table 1 and Fig. 4.
- Updated after the creation of the purchase order. An example is PR3 in Table 1.

Quantitative Metrics: in the considered case study, a significant number of purchase requisitions, 11.8%, have been modified after the creation of a corresponding purchase order.

Comparison versus Traditional Process Mining: Since many changes (corresponding to different activities) could be performed on a purchase requisition, the verification of this property starting from the traces of a traditional event log for the P2P process is problematic.

6.1.2 Application of the object interaction graph: detection of maintenance contracts

Problem Statement: purchase orders are usually placed to buy a set of products from a supplier. Therefore, they involve a limited number of invoices/payments. Some purchase orders are "maintenance contracts," which means they are used to cover periodic maintenance interventions of external companies. These maintenance contracts are left open and associated with the periodic invoices coming from such companies. This subclass of purchase orders can be detected thanks to the object interaction graph. In particular, orders that are associated with a significant amount of invoices are, in most cases, maintenance contracts. An example is PO3 in Table 1 and Fig. 2; in comparison with the other orders shown in Fig. 2, the number of invoices associated with the order is higher.
Quantitative Metrics: in the considered case study, 2.3% of the orders are associated with at least five different invoices.
Comparison versus Traditional Process Mining: while it is still possible to detect maintenance contracts on traditional P2P event logs, this depends on the chosen granularity and case notion. If the case notion is the purchase order and the granularity is at the document level, it is possible to count the number of invoices associated with each single purchase order. Otherwise, if the granularity is at the document item level, getting the actual number of invoices per order is complicated.

6.1.3 Application of the object creation graph: Maverick buying

Problem Statement: maverick buying is another workaround in the Procure-to-Pay process. To avoid a formal placement of the purchase order, the order is informally placed to the supplier, which sends the goods and the invoice to the company. After receiving the invoice, the purchase order is inserted into the ERP system. It is possible to use the object creation graph to detect maverick buying. What should happen in normal conditions is that the purchase order should be followed by an invoice. In contrast, maverick buying leads to invoices followed by purchase orders. An example is R7 in Table 1 and Fig. 4.
Quantitative Metrics: in the considered case study, the 1.6% of the orders suffer from maverick buying.
Comparison versus Traditional Process Mining: this problem is easier to detect on traditional P2P event logs, in comparison with the problems in the other subsections. However, if the invoices are also inserted using different transactions compared to the usual ones, the detection of this phenomenon can be problematic.

6.1.4 Application of the feature propagation: factors influencing invoice duration

The lifecycle of an invoice is directly influenced by the features of the purchase order and not by the invoice processing itself. Consider, for example, an invoice related to a purchase order suffering from a goods issue. This means that the items of the order are not delivered in a functioning state, and replacement may be sent. In addition, this influences the clearance (payment) of the invoice, with a payment block that could be inserted until the goods have arrived in a good state. In the object interaction graph (Fig. 2), we could see, for example, that invoice R6 is associated with a purchase order suffering from a goods issue. While the goods issue is not directly connected to the invoice, it could lead to a significant delay in the payment of the invoice.

6.2 Qualitative assessment on a synthetic event log

Input: a synthetic event log related to an order management process has been generated to support the reproducibility of the results of the techniques proposed in this paper. The event log is available at the address https://www.ocpm.info/synthetic_po.jsonocel, and its characteristics are described in Table 5. In this event log:

- Purchase orders are created with a "Create Purchase Order" activity, followed possibly by the "Change Price" activity and (except in the case of maverick buying) some invoices which are related to the purchase order.
- Invoices are created with a "Create Invoice" activity, followed possibly by some change activities ("Change Y1," "Change Y2," "Change Y3," "Change Y4',' and "Change Y5") and a "Payment" (with annexed object).
- In the case of maverick buying, the invoices are followed by a purchase order.

Table 5 Characteristics of the synthetic event log used for the qualitative assessment

Characteristic	Value
Number of events	1187
Number of objects	723
Number of activities	8
Number of object types	3

Anomaly Detection: different anomalies can be identified in the event log:

- Some invoices are related to a significantly bigger number of changes than other invoices.
- Some purchase orders are related to many invoices.

The event log can be directly uploaded to the OCPM tool www.ocpm.info. Then, the "isolation forests" component of the tool can be used to identify the most anomalous objects that appear on top of the list (see Fig. 5).

We see, for example, that the object with the identifier "PO_35" is associated with 24 different invoices, and that is exceptional enough that the object gets a very high anomaly score. The invoice with the identifier "INVOICE_149" is anomalous because many change activities ("Change Y1," "Change Y2," "Change Y3," "Change Y4," and "Change Y5") are executed. This is correctly identified by the tool.

Using the Results: An example application of the knowledge of anomalous objects is the filtering of the object-centric directly-follows graph. Figure 6 shows the object-centric directly-follows graph on the entire object-centric event log. Figure 7 shows the model computed on a filtered version of the log where we removed the 20% of most anomalous objects according to the isolation forest score. This can be done by selecting the *Advanced filtering* component of OCPM and executing the *Anomalous Objects Filtering (Isolation Forest)* component. We can see that some connections with a low number of occurrences have been filtered out.

6.3 Quantitative assessment using public available object-centric event logs

Goal: The qualitative assessment has shown that graph-based features are useful to detect complex conditions of interaction between different objects, both in the real P2P process and the synthetic event log, but a quantitative assessment (how much the features are useful in a generic setting?) is missing. In this subsection, we propose a quantitative assessment of the features extracted from our approach on top of more complex publicly available event logs. Indeed, we compare four different settings for feature extraction:

S1 (Baseline) only features related to the lifecycle of an object are considered. This is equivalent to performing feature extraction on top of traditional event logs.
S2 Features related to the lifecycle of an object and the object interaction graph are considered. This is equivalent to the setting described in [10].
S3 Features related to the lifecycle of an object and the object interaction and creation graph are considered.
S4 Features related to the lifecycle of an object and the object interaction, creation, and continuation graphs are considered.

Inputs: the logs that are used in the quantitative assessment are available at the address https://www.ocel-standard.org/ and their features are reported in Table 6. For some of the event logs, we filtered out the object types as specified in Table 6. This was done because objects of some object types were connected to a significant number of events, inducing a connection between most of the events and objects of the event log, therefore making graph-based feature extraction less effective. As an example, many orders contained the same products, so the *products* object type was filtered out from the "Order management log."

Technique-Dependent Considerations: we assessed the quality of features based on the quality of machine learning models that can be learned from the data. In particular, we want to predict the lifecycle duration starting from four different scenarios, S1-t, S2-t, S3-t, and S4-t, which are equivalent to the aforementioned scenarios with the exclusion of the lifecycle duration feature. For this, we use an extra trees regressor trained on 80% of the objects to predict the lifecycle duration of the remaining objects and compare the prediction with the actual value.

The effectiveness of the prediction was measured using two standard metrics (mean absolute percentage error (MAPE) and root of mean squared percentage error (RMSPE)), briefly reported below. Here, A_i is relative to the actual value (the effective throughput time) and F_i is relative to the predicted completion time.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right|$$

$$\text{RMSPE} = \frac{\sqrt{\sum_{i=1}^{n}(A_i - F_i)^2}}{n}$$

The results are reported in Table 7 (for the MAPE metric) and Table 8 (for the RMSPE metric). Lower values indicate better predictions, and, generally, the addition of the object interaction graph and the object creation graph contributes to improving the results of the prediction. Considering the MAPE metric, the best predictive results are obtained in the scenario S4-t for three of the four logs.

Therefore, considering several different graph-based features contributed to a better prediction in comparison with considering only features related to the lifecycle of an object (S1-t, which is equivalent to what is done in [6]) or just the object interaction graph (S2-t, which is equivalent to what is done in [10])

SQL explorer | Isolation Forest | Dimensionality Reduction | Decision Trees | Correlation Statistics | Conformance Rules

The isolation forest calculated on the objects' features permits to identify the objects with peculiar behavior (different from the one of the other objects).

Object ID	Object Type	Anomaly Score
PO_35	ORDER	0.5457392172964848
PO_35	ORDER	0.5255540300151416
PO_65	ORDER	0.5188066957828468
PO_23	ORDER	0.4699360446397309
PO_23	ORDER	0.46746750647783003
PO_23	ORDER	0.4665055664639341
PO_43	ORDER	0.4620189653136277
INVOICE_149	INVOICE	0.4448960259638731

Fig. 5 Isolation forests computed on the synthetic event log available at https://www.ocpm.info/synthetic_po.jsonocel in the machine learning component of the OCPM tool

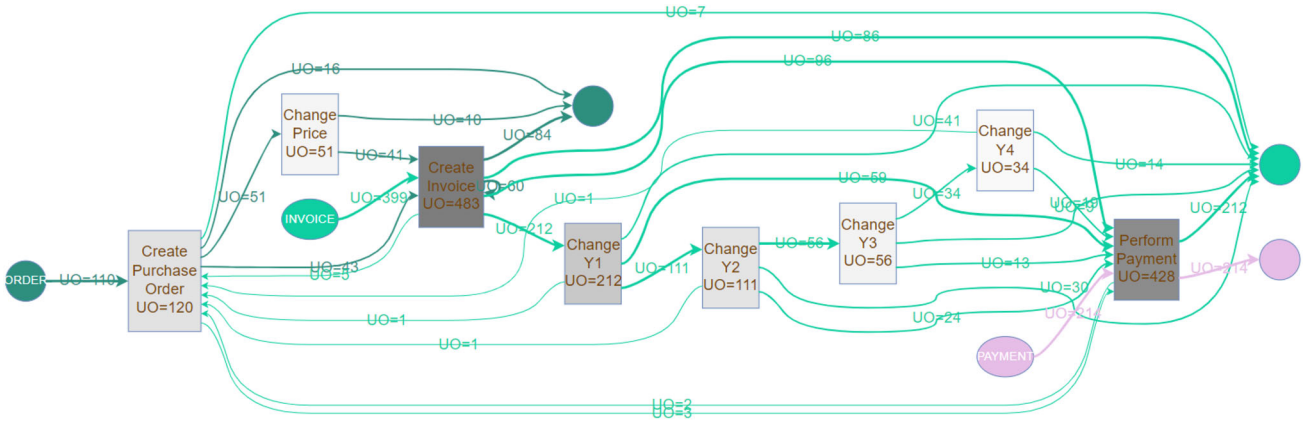

Fig. 6 Object-centric directly-follows graph [3] based on the object-centric event log available at the address https://www.ocpm.info/synthetic_po.jsonocel

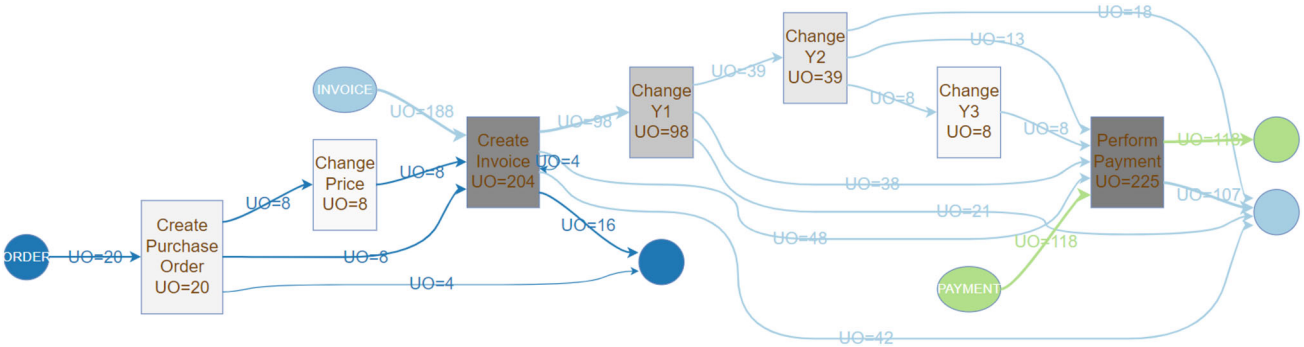

Fig. 7 Object-centric directly-follows graph [3] obtained filtering out the 20% of most anomalous objects. In comparison with the model discovered on the unfiltered log, this discovered model is significantly simpler

Table 6 Event logs used in the quantitative assessment

Event log	Cons. obj. types	N. events	N. objects	N. activities
Order management log	{Orders; items; packages}	22367	11484	11
SAP ERP IDES instance—O2C log	All	98350	107767	23
SAP ERP IDES instance—P2P log	{EBELN_EBELP; EBELN; BELNR_EBELN_EBELP; BELNR}	20554	62353	13
Recruiting process	{Applicants; applications; offers}	6607	1339	12

Table 7 Prediction error (MAPE) of the lifecycle duration in the four experimental scenarios S1-t, S2-t, S3-t, and S4-t (lower is better)

Event log	S1-t (%)	S2-t (%)	S3-t (%)	S4-t (%)
Order management log	**54**	55	55	55
SAP ERP IDES instance—O2C log	53	36	28	**27**
SAP ERP IDES instance—P2P log	147	134	132	**129**
Recruiting process	45	45	44	**43**

Bold highlights the value with minimum prediction error given the metric (i.e., MAPE)

Table 8 Prediction error (RMSPE) of the lifecycle duration in the four experimental scenarios S1-t, S2-t, S3-t, and S4-t (lower is better)

Event log	S1-t	S2-t	S3-t	S4-t
Order management log	**0.14D**	0.15D	0.15D	0.15D
SAP ERP IDES instance—O2C log	3.63D	3.14D	2.85D	**2.79D**
SAP ERP IDES instance—P2P log	0.14D	0.12D	**0.11D**	0.13D
Recruiting process	0.82D	0.82D	0.82D	0.82D

Bold highlights the value with minimum prediction error given the metric (i.e., MAPE)

7 Conclusion

In this paper, we introduce graph-based feature extraction on object-centric event logs. Many interesting applications of machine learning have been adapted to the process mining field. However, this could not be easily used in the object-centric setting since the quality of features extracted starting from the lifecycle of a single object is generally lower. Graph-based feature extraction techniques close the gap, allowing us to compute features based on the interaction between different objects.

The utility of the aforementioned features is highly dependent on the definition of an object-centric event log. Currently, in the OCEL standard, we do not consider the definition of any relationship between the objects, while this information could be already known during the log creation phase. Examples of relationships are the parent–children relation (for example, a purchase order is the "father" object of its purchase order items) and the bill of materials (a hierarchical structure including the final product, the list of raw materials, and the sub- and intermediate assemblies, with the final product at the top and the raw materials at the bottom). If the definition of object-centric event log is adapted to include relationships between objects, then the techniques of this paper are in part redundant. Moreover, the extraction of an object-centric event log could be quite arbitrary. In an ERP system, a purchase order is related to invoice(s) quite straightforwardly. However, when we create an object-centric event log, we can identify in different ways the events in which the relationship(s) occur(s). For example, the event with the activity "Create Purchase Order" could contain only the reference to the purchase order, while the event with the activity "Create Invoice" contains the reference to the purchase order and the invoice. Another possible choice is to place the information about the related invoices on the event with the activity "Create Purchase Order," while the event with the activity "Create Invoice" contains only the reference to the invoice. The number of arbitrary choices that are possible during the creation of an object-centric event log can determine a different behavior when extracting the features.

In this paper, we propose two tools (pm4py and OCPM) in which graph-based object-centric feature extraction techniques have been implemented. In particular, OCPM is web-based and offers some machine learning applications on top of the computed features. An assessment has been done on a real-life Procure-to-Pay event log, showing that graph-based features can be used to detect three different problems of the process. Some of these problems could not have been identified so easily on traditional event logs.

We leave out from the current version of the paper the definition of *situation table*. Indeed, the proposed graphs are built on top of the entire object-centric event log. While anomaly detection/clustering is possible starting from the features of the overall event log, some applications such as prescrip-

tive/predictive analytics need to consider a "prefix," which has not been done yet for object-centric event logs.

Acknowledgements We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

Author Contributions Alessandro Berti worked on the tool support, the experimental setting, and the first version of the text of the paper, while Johannes Herforth, Mahnaz Sadat Qafari, and Wil van der Aalst contributed to review the paper and improve its quality.

Funding Open Access funding enabled and organized by Projekt DEAL. This study was supported by Alexander von Humboldt (AvH) Stiftung.

Data Availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval Not applicable.

References

1. Adams, J.N., van der Aalst, W.M.P.: Ocπ: object-centric process insights. In: Bernardinello, L., Petrucci, L. (eds.) Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19–24, 2022, Proceedings, Lecture Notes in Computer Science, vol 13288. Springer, New York City, pp. 139–150 (2022). https://doi.org/10.1007/978-3-031-06653-5_8
2. Adams, J.N., Park, G., Levich, S., et al.: A framework for extracting and encoding features from object-centric event data. In: Troya, J., Medjahed, B., Piattini, M., et al. (eds.) Service-oriented computing, pp. 36–53. Springer, Cham (2022)
3. Berti, A., van der Aalst, W.M.P.: Extracting multiple viewpoint models from relational databases. In: Ceravolo, P., van Keulen, M., López, M.T.G. (eds.) Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13–14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers, Lecture Notes in Business Information Processing, vol 379. Springer, New York City, pp 24–51 (2019). https://doi.org/10.1007/978-3-030-46633-6_2
4. Cohen, W.W.: Fast effective rule induction. In: Prieditis, A., Russell, S. (eds.) Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9–12, 1995. Morgan Kaufmann, Massachusetts, USA, pp 115–123 (1995). https://doi.org/10.1016/b978-1-55860-377-6.50023-2
5. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Predictive performance monitoring of material handling systems using the performance spectrum. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24–26, 2019. IEEE, New York City, pp 137–144 (2019). https://doi.org/10.1109/ICPM.2019.00029
6. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016). https://doi.org/10.1016/j.is.2015.07.003
7. de Lima Bezerra F, Wainer J, van der Aalst, W.M.P.: Anomaly detection using process mining. In: Halpin TA, Krogstie, J., Nurcan, S., et al. (eds.) Enterprise, Business-Process and Information Systems Modeling, 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8–9, 2009. Proceedings, Lecture Notes in Business Information Processing, vol 29. Springer, New York City, pp 149–161 (2009). https://doi.org/10.1007/978-3-642-01862-6_13
8. Elkhovskaya, L., Kovalchuk, S.V.: Feature engineering with process mining technique for patient state predictions. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., et al. (eds.) Computational Science - ICCS 2021 - 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part III, Lecture Notes in Computer Science, vol 12744. Springer, pp 584–592 (2021). https://doi.org/10.1007/978-3-030-77967-2_48
9. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**(1–2), 109–141 (2021). https://doi.org/10.1007/s13740-021-00122-1
10. Galanti, R., de Leoni, M., Navarin, N., et al.: Object-centric process predictive analytics. Expert Syst. Appl. **213**, 119173 (2023). https://doi.org/10.1016/j.eswa.2022.119173
11. Ghahfarokhi, A.F., Park, G., Berti, A., et al.: OCEL: a standard for object-centric event logs. In: New Trends in Database and Information Systems - ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Tartu, Estonia, August 24–26, 2021, Proceedings, Communications in Computer and Information Science, vol 1450. Springer, New York City, pp 169–175 (2021). https://doi.org/10.1007/978-3-030-85082-1_16
12. Gherissi, W., Haddad, J.E., Grigori, D.: Object-centric predictive process monitoring. In: Troya J, Mirandola R, Navarro E, et al (eds) Service-Oriented Computing - ICSOC 2022 Workshops - ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29–December 2, 2022 Proceedings, Lecture Notes in Computer Science, vol 13821. Springer, pp 27–39 (2022). https://doi.org/10.1007/978-3-031-26507-5_3
13. Junior, S.B., Ceravolo, P., Damiani, E., et al.: Evaluating trace encoding methods in process mining. In: Bowles, J., Broccia, G., Nanni, M. (eds.) From Data to Models and Back - 9th International Symposium, DataMod 2020, Virtual Event, October 20, 2020, Revised Selected Papers, Lecture Notes in Computer Science, vol 12611. Springer, New York City, pp 174–189 (2020). https://doi.org/10.1007/978-3-030-70650-0_11
14. Klijn, E.L., Fahland, D.: Identifying and reducing errors in remaining time prediction due to inter-case dynamics. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4–9, 2020. IEEE, New York City, pp 25–32 (2020). https://doi.org/10.1109/ICPM49681.2020.00015
15. Mensi, A., Bicego, M.: A novel anomaly score for isolation forests. In: Ricci, E., Bulò, S.R., Snoek, C., et al. (eds.) Image Analysis and

Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9–13, 2019, Proceedings, Part I, Lecture Notes in Computer Science, vol 11751. Springer, New York City, pp 152–163 (2019). https://doi.org/10.1007/978-3-030-30642-7_14
16. Pourbafrani, M., van der Aalst, W.M.P.: Extracting process features from event logs to learn coarse-grained simulation models. In: Rosa, M.L., Sadiq, S.W., Teniente, E. (eds.) Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28–July 2, 2021, Proceedings, Lecture Notes in Computer Science, vol 12751. Springer, New York City, pp 125–140 (2021). https://doi.org/10.1007/978-3-030-79382-1_8
17. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Supporting decisions in production line processes by combining process mining and system dynamics. In: Ahram, T.Z., Karwowski, W., Vergnano, A., et al. (eds.) Intelligent Human Systems Integration 2020 - Proceedings of the 3rd International Conference on Intelligent Human Systems Integration (IHSI 2020): Integrating People and Intelligent Systems, February 19–21, 2020, Modena, Italy, Advances in Intelligent Systems and Computing, vol 1131. Springer, New York City, pp 461–467 (2020). https://doi.org/10.1007/978-3-030-39512-4_72
18. Pourbafrani, M., Kar, S., Kaiser, S., et al.: Remaining time prediction for processes with inter-case dynamics. In: Munoz-Gama, J., Lu, X. (eds.) Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31–November 4, 2021, Revised Selected Papers, Lecture Notes in Business Information Processing, vol 433. Springer, New York City, pp 140–153 (2021). https://doi.org/10.1007/978-3-030-98581-3_11
19. Qafari, M.S., van der Aalst, W.M.P.: Root cause analysis in process mining using structural equation models. In: del-Río-Ortega, A., Leopold, H., Santoro, F.M. (eds,) Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13–18, 2020, Revised Selected Papers, Lecture Notes in Business Information Processing, vol 397. Springer, New York City, pp 155–167 (2020). https://doi.org/10.1007/978-3-030-66498-5_12
20. Qafari, M.S., van der Aalst, W.M.P.: Case level counterfactual reasoning in process mining. In: Nurcan, S., Korthaus, A. (eds.) Intelligent Information Systems - CAiSE Forum 2021, Melbourne, VIC, Australia, June 28–July 2, 2021, Proceedings, Lecture Notes in Business Information Processing, vol 424. Springer, New York City, pp 55–63 (2021). https://doi.org/10.1007/978-3-030-79108-7_7
21. Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., et al.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9), 189:1-189:38 (2022). https://doi.org/10.1145/3472752
22. Tax, N., Verenich, I., Rosa, M.L., et al.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12–16, 2017, Proceedings, Lecture Notes in Computer Science, vol 10253. Springer, New York City, pp 477–492 (2017). https://doi.org/10.1007/978-3-319-59536-8_30
23. Tax, N., Teinemaa, I., van Zelst, S.J.: An interdisciplinary comparison of sequence modeling methods for next-element prediction. Softw. Syst. Model. **19**(6), 1345–1365 (2020). https://doi.org/10.1007/s10270-020-00789-3
24. van der Aalst, W.M.P.: Object-centric process mining: dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) Software Engineering and Formal Methods—17th International Conference, SEFM 2019, Oslo, Norway, September 18–20, 2019, Proceedings, Lecture Notes in Computer Science, vol 11724. Springer, New York City, pp 3–25 (2019). https://doi.org/10.1007/978-3-030-30446-1_1
25. Vazifehdoostirani, M., Genga, L., Dijkman, R.M.: Encoding high-level control-flow construct information for process outcome prediction. In: Burattin, A., Polyvyanyy, A., Weber, B. (eds.) 4th International Conference on Process Mining, ICPM 2022, Bolzano, Italy, October 23–28, 2022. IEEE, New York City, pp 48–55 (2022). https://doi.org/10.1109/ICPM57379.2022.9980737
26. Winter, K., Stertz, F., Rinderle-Ma, S.: Discovering instance and process spanning constraints from process execution logs. Inf. Syst. **89**, 101484 (2020). https://doi.org/10.1016/j.is.2019.101484